

| L Number | Hits | Search Text | DB | Time stamp |
|----------|------|---|--|------------------|
| 1 | 2 | DOM WITH OWNER | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:50 |
| 2 | 624 | DOM WITH DOCUMENT | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:50 |
| 3 | 53 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:51 |
| 4 | 2 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND THUMBNAIL | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:51 |
| 5 | 44 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:52 |
| 6 | 44 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND SUMMARY | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:52 |
| 7 | 10 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (SUMMARY WITH DOCUMENT) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:56 |
| 8 | 13 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (STREAM WITH DOCUMENT) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:56 |
| 9 | 3 | ((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (STREAM WITH DOCUMENT)) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (SUMMARY WITH DOCUMENT)) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:58 |
| 10 | 0 | DOWUMENT WITH STREAM | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:58 |
| 11 | 2520 | DOCUMENT WITH STREAM | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:58 |
| 12 | 4429 | DOCUMENT WITH MODEL | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 14:58 |
| 13 | 315 | (DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:00 |
| 14 | 46 | ((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:00 |
| 15 | 3 | ((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) AND THUMBNAIL | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:00 |
| 16 | 36 | ((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (SUMMARY WITH DOCUMENT)) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:01 |

| | | | | |
|----|----|--|--|------------------|
| 17 | 33 | (((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (STREAM WITH DOCUMENT)) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:01 |
| 18 | 33 | (((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (SUMMARY WITH DOCUMENT))) AND (((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (STREAM WITH DOCUMENT))) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:01 |
| 19 | 14 | (((((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (SUMMARY WITH DOCUMENT))) AND (((DOCUMENT WITH STREAM) AND (DOCUMENT WITH MODEL)) AND OWNER AND PERMISSION) NOT (((DOM WITH DOCUMENT) AND OWNER AND TYPE AND PERMISSION AND INDEX) AND STREAM) AND (STREAM WITH DOCUMENT)))) AND @AD<20001013 | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:01 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp | Comments | Error Definition | Errors |
|---|------|-----|------|---|--|----------------------|----------|------------------|--------|
| 1 | BRS | L1 | 2 | DOM WITH OWNER | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:50 | | | 0 |
| 2 | BRS | L2 | 624 | DOM WITH DOCUMENT | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:50 | | | 0 |
| 3 | BRS | L3 | 53 | 2 AND OWNER AND TYPE AND PERMISSION AND INDEX | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:51 | | | 0 |
| 4 | BRS | L4 | 2 | 3 AND THUMBNAİL | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:51 | | | 0 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp | Comments | Error Definition | Errors |
|---|------|-----|------|----------------------------------|--|----------------------|----------|------------------|--------|
| 5 | BRS | L5 | 44 | 3 AND STREAM | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:52 | | | 0 |
| 6 | BRS | L6 | 44 | 5 AND SUMMARY | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:52 | | | 0 |
| 7 | BRS | L7 | 10 | 5 AND (SUMMARY WITH DOCUMENT) | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:56 | | | 0 |
| 8 | BRS | L8 | 13 | 5 AND (STREAM WITH DOCUMENT) | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 14:56 | | | 0 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp | Comments | Error Definition | Errors |
|----|------|-----|------|----------------------|--|------------------|----------|------------------|--------|
| 9 | BRS | L9 | 3 | 8 NOT 7 | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 14:58 | | | 0 |
| 10 | BRS | L10 | 0 | DOCUMENT WITH STREAM | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 14:58 | | | 0 |
| 11 | BRS | L11 | 2520 | DOCUMENT WITH STREAM | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 14:58 | | | 0 |
| 12 | BRS | L12 | 4429 | DOCUMENT WITH MODEL | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 14:58 | | | 0 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp | Comments | Error Definition | Errors |
|----|------|-----|------|-----------------------------|--|------------------|----------|------------------|--------|
| 13 | BRS | L13 | 315 | 11 AND 12 | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:00 | | | 0 |
| 14 | BRS | L14 | 46 | 13 AND OWNER AND PERMISSION | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:00 | | | 0 |
| 15 | BRS | L15 | 3 | 14 AND THUMBNAIL | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:00 | | | 0 |
| 16 | BRS | L16 | 36 | 14 NOT 7 | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:01 | | | 0 |

| Type | L # | Hits | Search Text | DBs | Time Stamp | Comments | Error Definition | Errors |
|--------|-----|------|---------------------|--|----------------------|----------|------------------|--------|
| 17 BRS | L17 | 33 | 14 NOT 8 | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 15:01 | | | 0 |
| 18 BRS | L18 | 33 | 16 AND 17 | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 15:01 | | | 0 |
| 19 BRS | L19 | 14 | 18 AND @AD<20001013 | USPA T; US-P GPUB ; EPO; IBM_ TDB | 2004/03/0 9 15:01 | | | 0 |

| | U | 1 | Document ID | Issue Date | Pages | Title | Current OR | Current XRef | Retrieval Classif |
|----|-------------------------------------|--------------------------|-------------------|------------|-------|--|------------|-----------------------------------|-------------------|
| 1 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20030229529 A1 | 20031211 | 124 | Method for enterprise workforce planning | 705/8 | | |
| 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20030164856 A1 | 20030904 | 19 | Desktop, stream-based, information management system | 345/764 | 345/700 | |
| 3 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020120859 A1 | 20020829 | 97 | Method and apparatus for an improved security system mechanism in a business applications management system platform | 713/200 | 705/50 | |
| 4 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020073236 A1 | 20020613 | 96 | Method and apparatus for managing data exchange among systems in a network | 709/246 | 709/217 | |
| 5 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020073080 A1 | 20020613 | 99 | Method and apparatus for an information server | 707/3 | | |
| 6 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020055946 A1 | 20020509 | 165 | Enterprise, stream-based, information management system | 715/500 | | |
| 7 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020049788 A1 | 20020425 | 93 | Method and apparatus for a web content platform | 715/513 | 715/500 | |
| 8 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020049749 A1 | 20020425 | 97 | Method and apparatus for a business applications server management system platform | 707/3 | | |
| 9 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020049603 A1 | 20020425 | 92 | Method and apparatus for a business applications server | 705/1 | | |
| 10 | <input type="checkbox"/> | <input type="checkbox"/> | US 6643652 B2 | 20031104 | 88 | Method and apparatus for managing data exchange among systems in a network | 707/10 | 707/104.1; 709/202; 709/203 | |

app1

| | Inventor | S | C | P | 2 | 3 | 4 | 5 | Image Doc. Displayed | PT |
|----|---------------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------|--------------------------|
| 1 | Mui, Yet et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20030229529 | <input type="checkbox"/> |
| 2 | Prager, Randy et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20030164856 | <input type="checkbox"/> |
| 3 | Lipkin, Daniel S. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020120859 | <input type="checkbox"/> |
| 4 | Helgeson, Christopher S. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020073236 | <input type="checkbox"/> |
| 5 | Lipkin, Daniel S. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020073080 | <input type="checkbox"/> |
| 6 | Prager, Randy et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020055946 | <input type="checkbox"/> |
| 7 | Lipkin, Daniel S. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020049788 | <input type="checkbox"/> |
| 8 | Helgeson, Chris et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020049749 | <input type="checkbox"/> |
| 9 | Mehra, Gaurav et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020049603 | <input type="checkbox"/> |
| 10 | Helgeson, Christopher S. et al. | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6643652 | <input type="checkbox"/> |

| | U | 1 | Document ID | Issue Date | Pages | Title | Current OR | Current XRef | Retrieval Classif |
|---|-------------------------------------|--------------------------|-------------------------|------------|-------|--|------------|--------------|-------------------|
| 1 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20030179228 A1 | 20030925 | 114 | Instance browser for ontology | 345/738 | | |
| 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20030037181 A1 | 20030220 | 90 | Method and apparatus for providing process-container platforms | 719/328 | 709/202 | |
| 3 | <input type="checkbox"/> | <input type="checkbox"/> | US 20010047394 A1 | 20011129 | 68 | System, method, and computer program product for executing scripts on mobile devices | 709/217 | 709/230 | |

| | Inventor | S | C | P | 2 | 3 | 4 | 5 | Image Doc. Displayed | PT |
|---|---------------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------|--------------------------|
| 1 | Schreiber, Marcel Zvi et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20030179228 | <input type="checkbox"/> |
| 2 | Freed, Erik J. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20030037181 | <input type="checkbox"/> |
| 3 | Kloba, David D. et al. | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20010047394 | <input type="checkbox"/> |

| | U | 1 | Document ID | Issue Date | Pages | Title | Current OR | Current XRef | Retrieval Classif |
|----|-------------------------------------|---|------------------|------------|-------|--|------------|--|-------------------|
| 9 | <input checked="" type="checkbox"/> | | US 6269380 B1 | 20010731 | 16 | Property based mechanism for flexibility supporting front-end and back-end components having different communication protocols | 707/200 | 707/1; 707/10; 707/100; 707/104.1; 715/500; 715/514 | |
| 10 | <input checked="" type="checkbox"/> | | US 6266682 B1 | 20010724 | 14 | Tagging related files in a document management system | 715/501.1 | 707/5; 715/516 | |
| 11 | <input checked="" type="checkbox"/> | | US 6266670 B1 | 20010724 | 13 | User level accessing of low-level computer system operations. | 707/100 | 707/103R; 707/204; 715/514 | |
| 12 | <input checked="" type="checkbox"/> | | US 6253217 B1 | 20010626 | 16 | Active properties for dynamic document management system configuration | 715/500 | 713/1; 713/100 | |
| 13 | <input checked="" type="checkbox"/> | | US 6240429 B1 | 20010529 | 17 | Using attached properties to provide document services | 715/500 | | |
| 14 | <input checked="" type="checkbox"/> | | US 6192347 B1 | 20010220 | 198 | System and methods for computing to support decomposing property into separately valued components | 705/36 | 705/31; 705/35; 705/38 | |

| | Inventor | S | C | P | 2 | 3 | 4 | 5 | Image Doc. Displayed | PT |
|----|-------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------|--------------------------|
| 9 | Terry, Douglas B. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6269380 | <input type="checkbox"/> |
| 10 | LaMarca, Anthony G. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6266682 | <input type="checkbox"/> |
| 11 | LaMarca, Anthony G. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6266670 | <input type="checkbox"/> |
| 12 | Dourish, James P. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6253217 | <input type="checkbox"/> |
| 13 | Thornton, James D. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6240429 | <input type="checkbox"/> |
| 14 | Graff, Richard A. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6192347 | <input type="checkbox"/> |

| | U | 1 | Document ID | Issue Date | Pages | Title | Current OR | Current xRef | Retrieval Classif |
|---|-------------------------------------|--------------------------|-------------------|------------|-------|--|------------|--|-------------------|
| 1 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 20020055958 A1 | 20020509 | 15 | EXTENDING APPLICATION BEHAVIOR THROUGH ACTIVE PROPERTIES ATTACHED TO A DOCUMENT IN A DOCUMENT MANAGEMENT SYSTEM | 715/514 | | |
| 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6562076 B2 | 20030513 | 16 | Extending application behavior through active properties attached to a document in a document management system | 715/515 | 715/513; 717/100 | |
| 3 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6397231 B1 | 20020528 | 19 | Virtual documents generated via combined documents or portions of documents retrieved from data repositories | 715/515 | 707/102; 715/517; 715/522 | |
| 4 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6370553 B1 | 20020409 | 22 | Atomic and molecular documents | 715/514 | 707/10; 707/104.1 | |
| 5 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6345276 B1 | 20020205 | 14 | Representing base pointers in a shared memory heap | 707/100 | 707/101; 707/102; 709/214; 709/215; 711/100 | |
| 6 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6330573 B1 | 20011211 | 16 | Maintaining document identity across hierarchy and non-hierarchy file systems | 715/511 | 707/203 | |
| 7 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6324551 B1 | 20011127 | 15 | Self-contained document management based on document properties | 715/500 | 707/104.1 | |
| 8 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | US 6308179 B1 | 20011023 | 25 | User level controlled mechanism inter-positioned in a read/write path of a property-based document management system | 707/102 | 707/10; 707/104.1; 707/2; 707/4; 707/6; 707/8 | |

| | Inventor | S | C | P | 2 | 3 | 4 | 5 | Image Doc. Displayed | PT |
|---|---------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------|--------------------------|
| 1 | EDWARDS, WARREN K. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 20020055958 | <input type="checkbox"/> |
| 2 | Edwards, Warren K. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6562076 | <input type="checkbox"/> |
| 3 | Salisbury, Michael P. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6397231 | <input type="checkbox"/> |
| 4 | Edwards, Warren K. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6370553 | <input type="checkbox"/> |
| 5 | Lee, Henry | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6345276 | <input type="checkbox"/> |
| 6 | Salisbury, Michael P. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6330573 | <input type="checkbox"/> |
| 7 | Lamping, John O. et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6324551 | <input type="checkbox"/> |
| 8 | Petersen, Karin et al. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | US 6308179 | <input type="checkbox"/> |

| L Number | Hits | Search Text | DB | Time stamp |
|----------|------|--|--|------------------|
| 1 | 622 | "DOCUMENT OBJECT MODEL" | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:12 |
| 2 | 268 | "DOCUMENT OBJECT MODEL" AND STREAM | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:12 |
| 3 | 61 | ("DOCUMENT OBJECT MODEL" AND STREAM) AND (DOCUMENT WITH WINDOW) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:14 |
| 4 | 9 | ((("DOCUMENT OBJECT MODEL" AND STREAM) AND (DOCUMENT WITH WINDOW)) AND CURSOR AND SERVER | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:16 |
| 6 | 0 | ((("DOCUMENT OBJECT MODEL" AND STREAM) AND (DOCUMENT WITH WINDOW)) AND CURSOR AND SERVER) AND ((("DOCUMENT OBJECT MODEL" AND STREAM) AND (DOCUMENT WITH WINDOW)) AND @AD<20001013) | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:16 |
| 5 | 29 | ((("DOCUMENT OBJECT MODEL" AND STREAM) AND (DOCUMENT WITH WINDOW)) AND @AD<20001013 | USPAT; US-PGPUB; EPO; IBM_TDB | 2004/03/09 15:16 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp | Comments |
|---|------|-----|------|------------------------------|--|------------------|----------|
| 1 | BRS | L1 | 622 | "DOCUMENT OBJECT MODEL" | USPAT; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:12 | |
| 2 | BRS | L2 | 268 | 1 AND STREAM | USPAT; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:12 | |
| 3 | BRS | L3 | 61 | 2 AND (DOCUMENT WITH WINDOW) | USPAT; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:14 | |
| 4 | BRS | L4 | 9 | 3 AND CURSOR AND SERVER | USPAT; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:16 | |
| 5 | BRS | L6 | 0 | 4 AND 5 | USPAT; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:16 | |
| 6 | BRS | L5 | 29 | 3 AND @AD<20001013 | USPAT; US-P GPUB ; EPO; IBM_ TDB | 2004/03/09 15:16 | |

| | Error Definition | Er ro rs |
|---|---------------------|----------------|
| 1 | | 0 |
| 2 | | 0 |
| 3 | | 0 |
| 4 | | 0 |
| 5 | | 0 |
| 6 | | 0 |

 **Netscape®**
network

Search

DDOCUMENT MODEL STREAM

Go

enhanced by Google™

New! Search the Web for images of 'DDOCUMENT MODEL STREAM'

[Search Tips](#)

Did you mean: **DOCUMENT MODEL STREAM**

Matching Sites [About This](#)

Page: 1

1. [ixdom_hd.html](http://www.innoxmlp.com/doc/ixdom_hd.html)
... is the primary datatype for the entire Document Object Model. ... This is also the DDocument node used to create ... it out to the specified output stream (IxpStream). ...
http://www.innoxmlp.com/doc/ixdom_hd.html
2. [Compression of XML Data](http://www.cis.strath.ac.uk/~mathias/publications/MScThesis.pdf)
... The document is not any longer a stream of characters but a tree of specific objects. In this model, individual nodes must be accessible. ...
<http://www.cis.strath.ac.uk/~mathias/publications/MScThesis.pdf>
3. [High Performance Solutions for Business Document Automation and ...](http://www.isis-papyrus.com/Download/solutionscatalog.pdf)
... Fast development and the use of a Framework model where key decision criteria ... with OMR coding and inserting and the optimization of the mail stream for maximum ...
<http://www.isis-papyrus.com/Download/solutionscatalog.pdf>
4. [Table of Contents](http://brisbane.foe.org.au/pdf/CSA%20screen%20view.pdf)
... In the industrial model, large, highly capitalised farms are better able to withstand 'market fluctuations' and other risks of farming. ...
<http://brisbane.foe.org.au/pdf/CSA%20screen%20view.pdf>

Page: 1

DDOCUMENT MODEL STREAM

Go

enhanced by Google™

New! Search the Web for images of 'DDOCUMENT MODEL STREAM'

[Search Tips](#)



[Help](#) | [Terms of Service](#) | [Privacy Policy](#) | [Download Netscape 7.1](#)
[About Netscape Network](#) | Copyright © 2004 Netscape Communications Corp. All rights reserved.

define IXDOM_H

declared at: ixdom.h TBD

struct DNode

declared at: ixdom.h The DNode struct is the primary datatype for the entire Document Object Model. It represents a single node in the document tree. DNode provides a flat interface to manipulate any type of node, however, in certain cases some of the 'dom_...' functions can not be used. For example, nodes of TEXT_NODE type may not have children, and adding children to such nodes is an error. Each node has a node-name and may have a node-value. DNode-s of ELEMENT_NODE type can have also attributes. The following table lists the possible values:

| node-type | node-name | node-value | attributes |
|------------------------------------|---------------------------|-------------------------------------|------------------------|
| <u>ELEMENT_NODE</u> | tagName | NULL | <u>DOMNamedNodeMap</u> |
| <u>ATTRIBUTE_NODE</u> | name of attribute | value of attribute | NULL |
| <u>TEXT_NODE</u> | #text | content of the text node | NULL |
| <u>CDATA_SECTION_NODE</u> | #cdata-section | content of the CDATA Section | NULL |
| <u>ENTITY_REFERENCE_NODE</u> | name of entity referenced | NULL | NULL |
| <u>ENTITY_NODE</u> | entity name | NULL | NULL |
| <u>PROCESSING_INSTRUCTION_NODE</u> | target | entire content excluding the target | NULL |
| <u>COMMENT_NODE</u> | #comment | content of the comment | NULL |
| <u>DOCUMENT_NODE</u> | #document | NULL | NULL |
| <u>DOCUMENT_TYPE_NODE</u> | document type name | NULL | NULL |
| <u>DOCUMENT_FRAGMENT_NODE</u> | #document-fragment | NULL | NULL |
| <u>NOTATION_NODE</u> | notation name | NULL | NULL |

The following table lists the functions that can be applied for certain node-types. Note: the first parameter of these functions is always a pointer to a node to operate on.

| node-type | function |
|-----------|--|
| | <u>dom_getNodeName</u> <u>dom_getNodeValue</u> <u>dom_setNodeValue</u> <u>dom_getNodeType</u> <u>dom_getParentNode</u> <u>dom_getChildNodes</u> <u>dom_getFirstChild</u> |

| | |
|------------------------------------|--|
| Any type | <u>dom_getLastChild</u> <u>dom_getPreviousSibling</u> <u>dom_getNextSibling</u> <u>dom_getOwnerDocument</u> <u>dom_insertBefore</u> <u>dom_replaceChild</u> <u>dom_removeChild</u> <u>dom_appendChild</u> <u>dom_hasChildNodes</u> <u>dom_cloneNode</u> |
| <u>ELEMENT_NODE</u> | <u>dom_getTagName</u> <u>dom_getAttributes</u> <u>dom_getAttribute</u> <u>dom_setAttribute</u> <u>dom_removeAttribute</u> <u>dom_getAttributeNode</u> <u>dom_setAttributeNode</u> <u>dom_removeAttributeNode</u> <u>dom_normalize</u> <u>dom_getElementsByTagName</u> |
| <u>ATTRIBUTE_NODE</u> | <u>dom_getAttrName</u> <u>dom_getAttrValue</u> <u>dom_setAttrValue</u> <u>dom_attrIsSpecified</u> |
| <u>TEXT_NODE</u> | <u>dom_splitText</u> <u>dom_getCharacterData</u> <u>dom_getDataLength</u> <u>dom_substringData</u> <u>dom_appendData</u> <u>dom_insertData</u> <u>dom_deleteData</u> <u>dom_replaceData</u> |
| <u>CDATA_SECTION_NODE</u> | <u>dom_getCharacterData</u> <u>dom_getDataLength</u> <u>dom_substringData</u> <u>dom_appendData</u> <u>dom_insertData</u> <u>dom_deleteData</u> <u>dom_replaceData</u> |
| <u>ENTITY_REFERENCE_NODE</u> | --- |
| <u>ENTITY_NODE</u> | <u>dom_getPublicId</u> <u>dom_getSystemId</u> |
| <u>PROCESSING_INSTRUCTION_NODE</u> | <u>dom_getPiTarget</u> <u>dom_getPiData</u> <u>dom_setPiData</u> |
| <u>COMMENT_NODE</u> | <u>dom_getCharacterData</u> <u>dom_getDataLength</u> <u>dom_substringData</u> <u>dom_appendData</u> <u>dom_insertData</u> <u>dom_deleteData</u> <u>dom_replaceData</u> |
| | <u>dom_setImplementation</u> |

| | |
|-------------------------------|--|
| <u>DOCUMENT_NODE</u> | <u>dom_getImplementation</u> <u>dom_getDoctype</u> <u>dom_getDocumentElement</u> <u>dom_createElement</u> <u>dom_createDocumentFragment</u> <u>dom_createTextNode</u> <u>dom_createComment</u> <u>dom_createCDATASection</u> <u>dom_createProcessingInstruction</u> <u>dom_createAttribute</u> <u>dom_createEntityReference</u> <u>dom_getElementsByTagName</u> |
| <u>DOCUMENT_TYPE_NODE</u> | <u>dom_getDoctypeName</u> <u>dom_getEntities</u> <u>dom_getNotations</u> <u>dom_getPublicId</u> <u>dom_getSystemId</u> |
| <u>DOCUMENT_FRAGMENT_NODE</u> | --- |
| <u>NOTATION_NODE</u> | <u>dom_getPublicId</u> <u>dom_getSystemId</u> |

struct DNodeList

declared at: ixdom.h DNodeList provides an ordered collection of nodes. The items in the DNodeList are accessible via an integral index, starting from 0, using dom_getListItem(). dom_getListLength() returns the number of nodes in the DNodeList object. See also: dom_getChildNodes(), dom_getElementsByTagName().

struct DNamedNodeMap

declared at: ixdom.h DNamedNodeMap represents a collections of nodes that can be accessed by name. Note that DNamedNodeMap is not maintained in any particular order. DNode-s contained in an DNamedNodeMap object may also be accessed by an ordinal index, but this is simply to allow convenient enumeration of the contents of a DNamedNodeMap. See also: dom_getAttributes(), dom_getEntities(), dom_getNotation().

type DImplementation

declared at: ixdom.h A void pointer that can be attached to a document node. The C implementation does not use this value.

struct DString

declared at: ixdom.h This string type is exclusively used by the DOM (C) module for internal representation, and to provide return values. These strings are owned by the nodes, hence applications must not free them.

- length** The number of XmlChar-s that precede the terminating 0 character. Note: the terminating zero is also an XmlChar, with zero value.
- buf** The memory buffer of the string. The size of the allocated memory for a DString object is calculated at creation time, to give enough room for the entire string, plus for the terminating zero.

struct DMonitorData

declared at:[ixdom.h](#) DMonitorData gives the context, when a certain event occurred during building the tree. This type of object is passed to a monitoring callback-function. Typically it can be used for error reporting. See also [dom_build\(\)](#) and [DMonitor](#).

client_data The same 'client_data' pointer that was given to [dom_build\(\)](#).

ptoken This is the current token of the low-level XML parser. See also [IxpCBData](#), [ixpParse\(\)](#).

str The current string that belongs to the current 'ptoken'. This may be the context of the error, or the name of an element, etc.

slen The length of the 'str'.

document The document node that is being to build.

parent The parent node of the current node that is being to build.

function type DMonitor

declared at:[ixdom.h](#) If the application gives a monitor (a call-back function) to the [dom_build\(\)](#) function, then the monitor will be called-back if an error occurs during the parsing. The monitor's argument (DMonitorData) provides the context of the error. Additionally to errors, other low-level events may be reported as well. See also [dom_build\(\)](#).

function type DErrorHandler

declared at:[ixdom.h](#) An error handler function can be associated to an existing DOM tree. If an error occurs during the tree manipulation, the function is called. In contrast to C++ exception handling, the handler can safely return, the tree remains in a consistent state. See also [dom_build\(\)](#) and [dom_setErrorHandler\(\)](#).

function type DVisitor

declared at:[ixdom.h](#) The application should implement this type of function in order to traverse the tree. The [dom_traverseTree\(\)](#) function takes a DVisitor function, and calls-back for each node during walking on the tree.

function dom_createDocument

declared at:[ixdom.h](#) It creates and returns a new DOM tree (document node) that can be populated by the application. The application owns the newly created tree, so it is responsible to release it with [dom_destroyDocument\(\)](#).

function dom_destroyDocument

declared at:[ixdom.h](#) The application should call this function to release the entire document tree.

function dom_setImplementation

declared at:[ixdom.h](#) Assign a [DImplementation](#) to a DOM tree. This is a helper function for the C++ implementation, the C module does not use it for any purpose.

function dom_getImplementation

declared at:[ixdom.h](#) Retrieve the [DImplementation](#) that was assigned by [dom_setImplementation](#).

function dom_getDoctype

declared at:[ixdom.h](#) Returns the DocumentType node that is associated to the tree.

function dom_getDocumentElement

declared at:[ixdom.h](#) Returns the root element of the document tree.

function dom_createElement

declared at:[ixdom.h](#) Creates and returns an Element node, with the specified name.

function dom_createDocumentFragment

declared at:[ixdom.h](#) Creates an empty DocumentFragment node.

function dom_createTextNode

declared at:[ixdom.h](#) Creates a Text node, given the specified string.

function dom_createComment

declared at:[ixdom.h](#) Creates a Comment node given the specified string.

function dom_createCDATASection

declared at:[ixdom.h](#) Creates a CDATASection node whose value is the specified string.

function dom_createProcessingInstruction

declared at:[ixdom.h](#) Creates a ProcessingInstruction node given the specified target and data strings.

function dom_createAttribute

declared at:[ixdom.h](#) Creates an Attr node of the given name.

function dom_createEntityReference

declared at:[ixdom.h](#) Creates an EntityReference node given the specified name.

function dom_getElementsByTagName

declared at:[ixdom.h](#) Returns a [DNodeList](#) of all the Elements node with a given tag name in the order in which they would be encountered in a preorder traversal of the document tree.

function dom_getNodeName

declared at:[ixdom.h](#) Returns the name of the node.

function dom_getNodeValue

declared at:[ixdom.h](#) Returns the value of the node.

function dom_setNodeValue

declared at:[ixdom.h](#) Sets the value of the node. It is an error ([DOM_NO_MODIFICATION_ALLOWED_ERR](#)) to attempt to modify a readonly value, example "#text".

function dom_getNodeType

declared at:[ixdom.h](#) Returns the type of the node.

function dom_getParentNode

declared at:[ixdom.h](#) Returns the parent of this node. All nodes, except Document, DocumentFragment, and Attr may have a parent. However, if a node has just been created and not yet added to the tree, or if it has been removed from the tree, then the function returns NULL.

function dom_getChildNodes

declared at:[ixdom.h](#) Returns a [DNodeList](#) that contains all children of this node. If there are no children, then the [DNodeList](#) does not contain nodes. The content of the returned [DNodeList](#) is "live" in the sense that, for instance, changes to the children of the node object that it was created from are immediately reflected in the nodes returned by the [NodeList](#) accessors; it is not a static snapshot of the content of the node. This is true for every [DNodeList](#), including the ones returned by the [dom_getElementsByTagName](#) function.

function dom_getFirstChild

declared at:[ixdom.h](#) Returns the first child of this node. If there is no such node, the function returns NULL.

function dom_getLastChild

declared at:[ixdom.h](#) Returns the last child of this node. If there is no such node, the function returns NULL.

function dom_getPreviousSibling

declared at: ixdom.h Returns the node immediately preceding this node. If there is no such node, the function returns NULL.

function dom_getNextSibling

declared at: ixdom.h Returns the node immediately following this node. If there is no such node, the function returns NULL.

function dom_getAttributes

declared at: ixdom.h Returns a DNamedNodeMap containing the attributes of this node (if it is an Element) or NULL otherwise.

function dom_getOwnerDocument

declared at: ixdom.h Returns the Document node associated with this node. This is also the DDocument node used to create new nodes. When this node is a Document node, then it returns NULL.

function dom_insertBefore

declared at: ixdom.h dom_insertBefore(this,newChild,refChild) inserts the node newChild before the existing child node refChild. If refChild is NULL, then it inserts newChild at the end of the list of children. If newChild is a DocumentFragment node, then all of its children are inserted, in the same order, before refChild. If the newChild is already in the tree, it is first removed. The function returns the node being inserted. The following errors may occur: DOM_HIERARCHY_REQUEST_ERR, DOM_WRONG_DOCUMENT_ERR, DOM_NO_MODIFICATION_ALLOWED_ERR, DOM_NOT_FOUND_ERR.

function dom_replaceChild

declared at: ixdom.h dom_replaceChild(this,newChild,oldChild) replaces the child node oldChild with newChild in the list of children, and returns the oldChild node. If the newChild is already in the tree, it is first removed. The following errors may occur: DOM_HIERARCHY_REQUEST_ERR, DOM_WRONG_DOCUMENT_ERR, DOM_NO_MODIFICATION_ALLOWED_ERR, DOM_NOT_FOUND_ERR.

function dom_removeChild

declared at: ixdom.h Removes the specified child node from the list of children, and returns it. The following errors may occur: DOM_NO_MODIFICATION_ALLOWED_ERR, DOM_NOT_FOUND_ERR.

function dom_appendChild

declared at: ixdom.h Adds the specified new child to the end of the list of children of this node. If the new child is already in the tree, it is first removed. The function returns the node added. The following errors may occur: DOM_HIERARCHY_REQUEST_ERR, DOM_WRONG_DOCUMENT_ERR, DOM_NO_MODIFICATION_ALLOWED_ERR.

function dom_hasChildNodes

declared at: [ixdom.h](#) Returns TRUE, if the node has children, FALSE if not.

function dom_cloneNode

declared at: [ixdom.h](#) Returns a duplicate of this node. The duplicate node has no parent ([dom_parentNode\(\)](#) returns NULL). Cloning an Element copies all attributes and their values, but this method does not copy any text it contains unless it is a deep clone, since the text is contained in a child DOMText node. Cloning any other type of node simply returns a copy of this node.

function dom_getListItem

declared at: [ixdom.h](#) [dom_getListItem\(list,idx\)](#) returns the idx-th item in the collection. If idx is greater than or equal to the number of nodes in the list, then the function returns NULL.

function dom_getListLength

declared at: [ixdom.h](#) Returns the number of nodes in the list.

function dom_getNamedItem

declared at: [ixdom.h](#) [dom_getNamedItem\(nodemap,name\)](#) returns a DNode object with the specified 'name', or NULL if the specified name did not identify any node in the map.

function dom_setNamedItem

declared at: [ixdom.h](#) Adds a DNode object to the map, using its name. Nodes of certain types have 'fix' name e.g. "#text", hence they can not be stored as the names would clash.

function dom_removeNamedItem

declared at: [ixdom.h](#) [dom_removeNamedItem\(nodemap,name\)](#) removes the node specified by 'name' from the map, and returns it. If the specified name did not identify any node in the map, then it returns NULL.

function dom_getMapItem

declared at: [ixdom.h](#) [dom_getMapItem\(nodemap,idx\)](#) returns the idxth item in the map. If index is greater than or equal to the number of nodes in the map, then the function returns NULL.

function dom_getMapLength

declared at: [ixdom.h](#) Returns the number of nodes in the map.

function dom_getCharacterData

declared at: [ixdom.h](#) Returns the character data of the node.

function dom_getDataLength

declared at: ixdom.h Returns the number of character units that are available through dom_getCharacterData() and the dom_substringData() function below. This may have the value zero, i.e., node may be empty. Note: the character unit is defined by XmlChar.

function dom_substringData

declared at: ixdom.h dom_substringData(this,offset,count) returns the 'count' character substring of the node's character data, starting at 'offset'. If the sum of 'offset' and 'count' exceeds the length, then all characters to the end of the data are returned. The following error may occur: DOM_INDEX_SIZE_ERR.

function dom_appendData

declared at: ixdom.h Appends the specified string to the end of the character data of the node. Upon success, dom_getData() provides access to the concatenation of data and the string specified.

function dom_insertData

declared at: ixdom.h dom_insertData(this,offset,str) inserts 'str' at 'offset'.

function dom_deleteData

declared at: ixdom.h dom_deleteData(this,offset,count) removes 'count' characters, starting at 'offset'. If the sum of 'offset' and 'count' exceeds length of the character data of the node, then all characters from 'offset' to the end of the data are deleted.

function dom_replaceData

declared at: ixdom.h dom_replaceData(this,offset,count,str) replaces 'count' characters starting at 'offset' by str. If the sum of 'offset' and 'count' exceeds length, then all characters to the end of the data are replaced.

function dom_getAttrName

declared at: ixdom.h Returns the name of the attribute.

function dom_getAttrValue

declared at: ixdom.h Returns the value of the attribute.

function dom_setAttrValue

declared at: ixdom.h Sets the value of the attribute.

function dom_attrIsSpecified

declared at: ixdom.h Returns TRUE, if this attribute was explicitly given a value in the original XML document, otherwise it returns FALSE.

function dom_getTagName

declared at:[ixdom.h](#) Returns the name of the element node (tagName).

function dom_getAttribute

declared at:[ixdom.h](#) Retrieves an attribute by its name, and returns the corresponding attribute value. If there is no attribute with the specified name, then it returns NULL.

function dom_setAttribute

declared at:[ixdom.h](#) [dom_setAttribute\(this,name,value\)](#) adds a new attribute with 'name', which has the specified value. If an attribute with that name is already present in the element, its value is changed to be that of the value parameter. This value is a simple string, it is not parsed as it is being set. So any markup (such as syntax to be recognized as an entity reference) is treated as literal text, and needs to be appropriately escaped by the application when it is written out. Note: [dom_printTree\(\)](#) makes the proper escaping.

function dom_removeAttribute

declared at:[ixdom.h](#) Removes an attribute by the specified name.

function dom_getAttributeNode

declared at:[ixdom.h](#) Returns a DOMAttribute node by the specified name, or NULL if there is no such attribute.

function dom_setAttributeNode

declared at:[ixdom.h](#) Adds a new attribute. If an attribute with that name is already present in the element, it is replaced by the new one. If the specified attribute replaces an existing attribute with the same name, then the old Attr node is returned, otherwise NULL is returned. Note: The DOM user must explicitly clone Attr nodes to re-use them in other elements. The following errors may occur:
[DOM_WRONG_DOCUMENT_ERR](#), [DOM_NO_MODIFICATION_ALLOWED_ERR](#),
[DOM_INUSE_ATTRIBUTE_ERR](#).

function dom_removeAttributeNode

declared at:[ixdom.h](#) Removes and returns the specified [DOMAttr](#) node. The function returns NULL, if the specified attribute node does not belong this element node. The following error may occur:
[DOM_NOT_FOUND_ERR](#).

function dom_normalize

declared at:[ixdom.h](#) Puts all Text nodes in the full depth of the sub-tree underneath this Element into a "normal" form where only markup (e.g., tags, comments, processing instructions, CDATA sections, and entity references) separates Text nodes, i.e., there are no adjacent Text nodes. This can be used to ensure that the DOM view of a document is the same as if it were saved and re-loaded, and is useful when operations (such as XPointer lookups) that depend on a particular document tree structure are to be used.

function dom_splitText

declared at:[ixdom.h](#) Breaks this Text node into two Text nodes at the specified offset. The character data of this node is splitted into two strings. The first part (up to the offset point) remains in this node, whereas the second part (from the offset point) is moved into the new Text node. The new node immediately inserted as the next sibling of this node. The function returns the new node.

function dom_getDoctypeName

declared at:[ixdom.h](#) Returns the name of the document type; i.e., the name that immediately follos the DOCTYPE keyword in the textual XML document.

function dom_getEntities

declared at:[ixdom.h](#) Returns a [DNamedNodeMap](#) node, which contains the collection of Entity nodes, defined for the document. Note: the XML parser expands the internal entities, thus they do not appear in the returned map.

function dom_getNotations

declared at:[ixdom.h](#) Returns a [DNamedNodeMap](#) node, which contains the collection of Notation nodes, defined for the document.

function dom_getPublicId

declared at:[ixdom.h](#) Returns the public identifier of the node, if it has one.

function dom_getSystemId

declared at:[ixdom.h](#) Returns the system identifier of the node, if it has one.

function dom_getNotationName

declared at:[ixdom.h](#) Returns the name of the notation that is associated to this entity.

function dom_getPiTarget

declared at:[ixdom.h](#) Returns the target of this processing instruction. E.g.: <**target** data part?>

function dom_getPiData

declared at:[ixdom.h](#) Returns the data part of this processing instruction. E.g.: <target **data part**?>

function dom_setPiData

declared at:[ixdom.h](#) Sets the data part of the processing instruction.

function dom_printTree

declared at: ixdom.h dom_printTree() serializes a full or sub-tree that is given as the first parameter, and prints it out to the specified output stream (IxpStream). The output character encoding is the same as the encoding of the tree.

function dom_traverseTree

declared at: ixdom.h The function traverses the tree/subtree denoted by the first parameter (tree). While traversing, the 'visit' function (second parameter) is called for each node. The third parameter indicates, when to call the 'visit' function. If it has the value of VISIT_WHEN_ENTER, then it calls 'visit', when the node appears at the first time. If it has the VISIT_WHEN_LEAVE, then it calls only after that its children have been visited. If it has the VISIT_ENTER_AND_LEAVE value, then it calls both times. The function returns FALSE, if the 'visit' function has aborted the traversing, otherwise TRUE.

function dom_build

declared at: ixdom.h dom_build() creates a new DOM tree (document node) and populates the tree, from the XML document that is given by the IxpParams argument. If a monitor (DMonitor) is provided by the application, then it will be called-back for each error and other low-level events, which were specified by 'event_types' parameter. If a 'client_data' is also provided, then it will be passed to the monitor function. The 'client_data' (IxpClientData) may point to an application's data structure, to account the occurred errors. In case of fatal error the monitor is called, then dom_build() returns a NULL pointer.

function dom_setErrorHandler

declared at: ixdom.h This function attaches an error-handler (DErrorHandler) to an existing tree (document node). This handler will be called whenever an error occurs during the tree manipulation. The 'client_data' parameter (IxpClientData) will be passed to the handler function.